3/30/2021
Ken Amamori
CS 4375

# Concurrency Beyond Locks Assignment Report

## Instruction:

Time your fixed-up version of pump.py (or pump-fixed.py above) with a buffer sizes of 1 and 100.
Now modify it to use semaphores, with release() and acquire(), or to use condition variables, with wait() and notify().
Predict whether the new version will be faster --- specifically whether it will require less real time, less user time, and less system time --- and explain why.
Time your new version.
Were your predictions correct?
If not, how can you explain what the results?

## Explanation:

Prediction: New version will be faster for real, user, will less time, and sys time will be about the same. The reason is because the original code only uses one variable to handle threads. Inside of the while loop, it repeatedly releases and acquires. When while loop breaks, it have to go to the last line “bufLock.release()” in order to unlock the lock and wake the other thread. By using condition variable, inside the loop will be redundant and with condition variable, notify() can be useful since it wakes up other thread.

In order to modified the given file pump-fixed.py I used condition variable. With condition variable, I used methods stated in instructions, wait() and notify().

Firstly, I introduced the variable that will be needed for condition variable. Imported Condition from threading and introduced two aditinal variable to the program. charAdded and charRemove. Both varaibale assigned Condition(bufLock) where bufLock is a lock.

Secondary, I modified inside of while-loop on both methods, putChar and getChar. I switched both lines of code, “bufLock.release()” and “bufLock.acquire()”, to charAdded.wait() – in putChar or charRemoved.wait() – in getChar() method.

Third, Since there are wait()s, now it is time to put notify() which handles waking up other threads. Conditional variable only works when the lock is being locked, thus, I put nofity() methods right before bufLock is released on both methods, getChar() and putChar().

Lastly, I added charRemoved and charAdded variable to global variable.

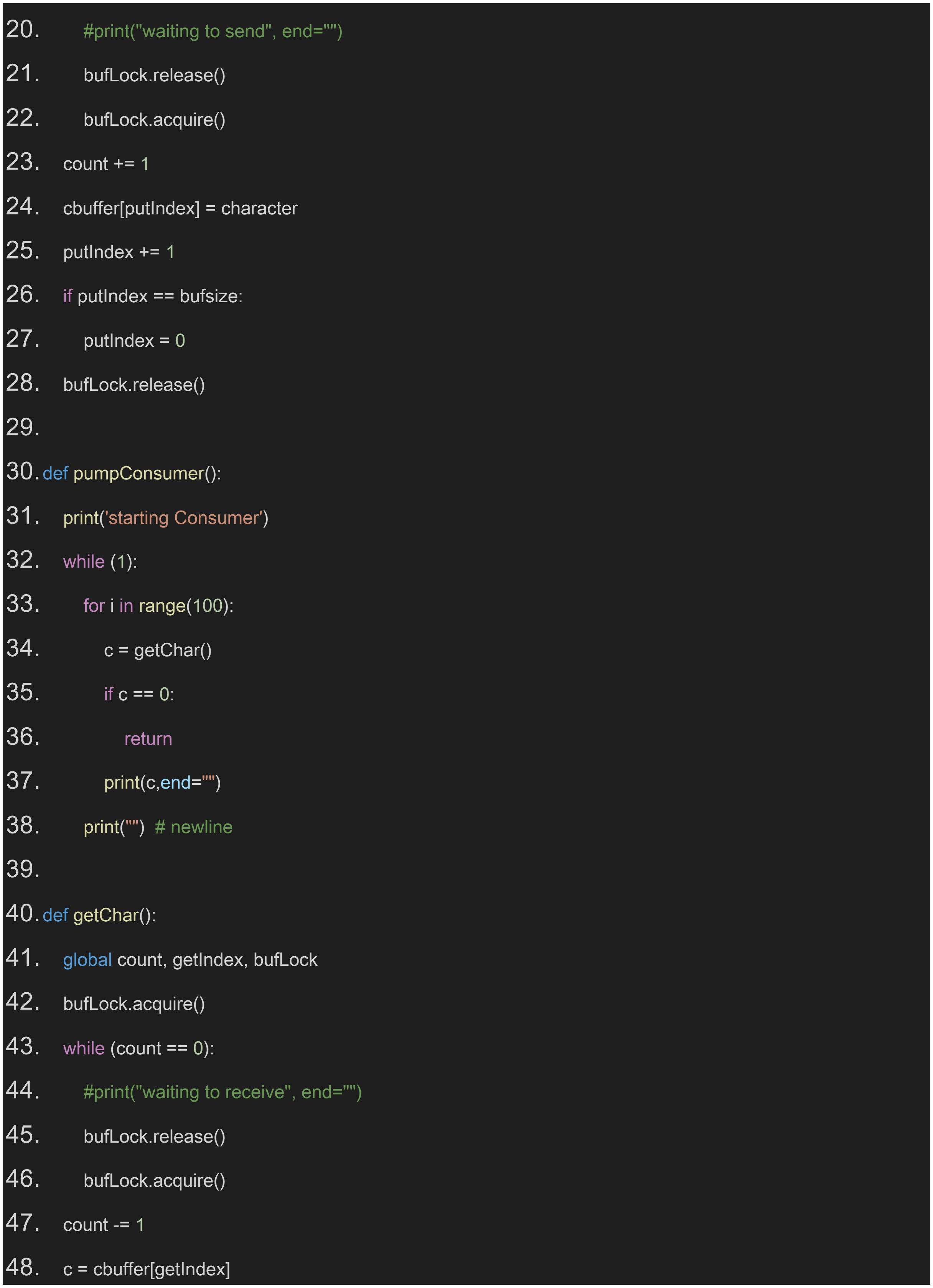
### Result:

The result was partially expected, and the other part was wrong. Real, user time was reduced with modified version of code as expected. In addition, sys time marked less time by half comparing to original code. Sys is the amount of CPU time that kernel spend within the processes, in other words, the executing CPU time spent in system calls within the kernel. The original version uses one lock to handle every action in program which was reduced in modified version due to the introduction of two additional variable. Within loop, it only wait() meaning sleeping until the other wakes up the sleeping condition variable. Furthermore, in order to run other action (producer -> consumer, or consumer -> producer), it will be handle by charAdded and charRemoved which is more efficient as the processes running in kernel.

# Code:

## Pump-fixed.py (Not modified):

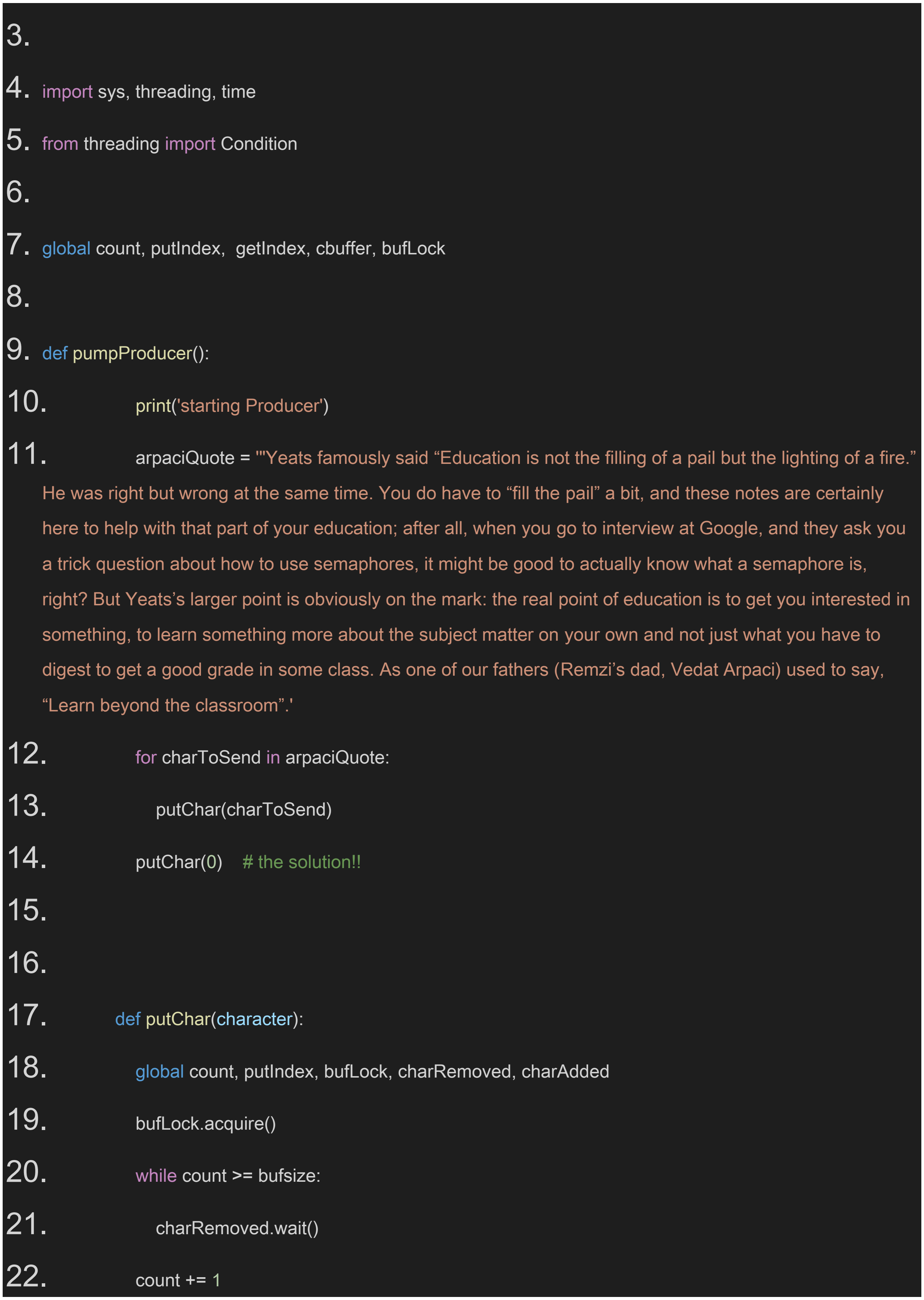
```
#!/usr/bin/env python3
# pump.py  Nigel Ward, 2021.  Based on John Osterhout's Producer/Consumer code.

import sys, threading, time

global count, putIndex,  getIndex, cbuffer, bufLock

def pumpProducer():
    print('starting Producer')
    arpaciQuote = '''Yeats famously said “Education is not the filling of a pail but the lighting of a fire.” He was right but wrong at the same time. You do have to “fill the pail” a bit, and these notes are certainly here to help with that part of your education; after all, when you go to interview at Google, and they ask you a trick question about how to use semaphores, it might be good to actually know what a semaphore is, right? But Yeats’s larger point is obviously on the mark: the real point of education is to get you interested in something, to learn something more about the subject matter on your own and not just what you have to digest to get a good grade in some class. As one of our fathers (Remzi’s dad, Vedat Arpaci) used to say, “Learn beyond the classroom”.'
    for charToSend in arpaciQuote:
        putChar(charToSend)
    putChar(0)    # the solution!!


def putChar(character):
    global count, putIndex, bufLock
    bufLock.acquire()
    while count >= bufsize:
```



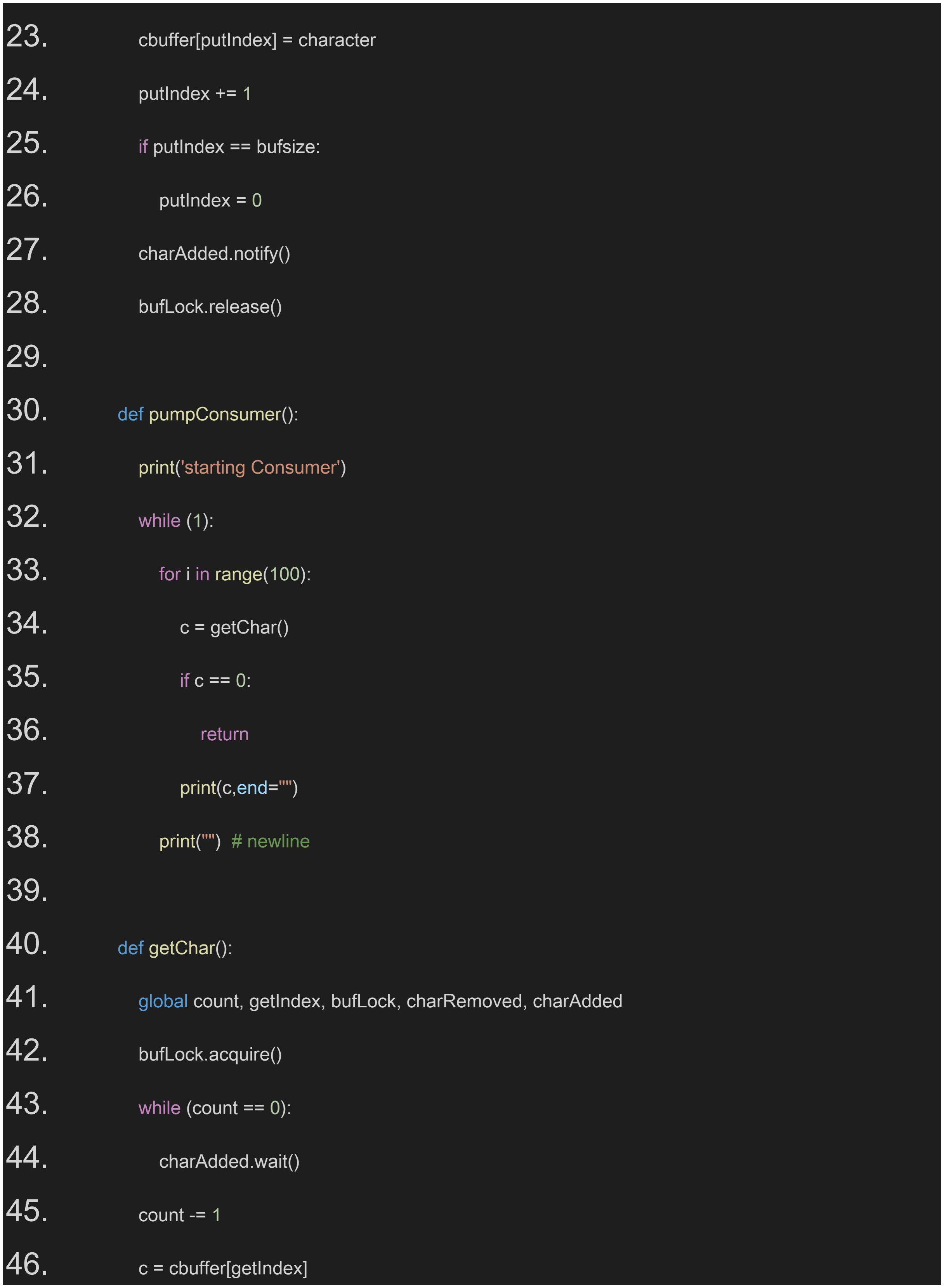
```
        #print("waiting to send", end="")
        bufLock.release()
        bufLock.acquire()
    count += 1
    cbuffer[putIndex] = character
    putIndex += 1
    if putIndex == bufsize:
        putIndex = 0
    bufLock.release()

def pumpConsumer():
    print('starting Consumer')
    while (1):
        for i in range(100):
            c = getChar()
            if c == 0:
                return
            print(c,end="")
        print("")  # newline

def getChar():
    global count, getIndex, bufLock
    bufLock.acquire()
    while (count == 0):
        #print("waiting to receive", end="")
        bufLock.release()
        bufLock.acquire()
    count -= 1
    c = cbuffer[getIndex]
    getIndex += 1
    if (getIndex == bufsize):
        getIndex = 0
    bufLock.release()
    return c

### main ###
if len(sys.argv) != 2:
    print(len(sys.argv))
    print("usage: pump bufferSize")
    exit(1)

bufsize = int(sys.argv[1])
cbuffer = ['x'] * bufsize    # circular buffer; x means uninitialized
count = putIndex = getIndex = 0
bufLock = threading.Lock()

consumer = threading.Thread(target=pumpConsumer)
consumer.start()

producer = threading.Thread(target=pumpProducer)
producer.start()

producer.join()

```

Pump-fixed-modifed.py (Modified):

```
#!/usr/bin/env python3
# pump.py  Nigel Ward, 2021.  Based on John Osterhout's Producer/Consumer code.
```



```

import sys, threading, time
from threading import Condition

global count, putIndex,  getIndex, cbuffer, bufLock

def pumpProducer():
        print('starting Producer')
        arpaciQuote = '''Yeats famously said “Education is not the filling of a pail but the lighting of a fire.” He was right but wrong at the same time. You do have to “fill the pail” a bit, and these notes are certainly here to help with that part of your education; after all, when you go to interview at Google, and they ask you a trick question about how to use semaphores, it might be good to actually know what a semaphore is, right? But Yeats’s larger point is obviously on the mark: the real point of education is to get you interested in something, to learn something more about the subject matter on your own and not just what you have to digest to get a good grade in some class. As one of our fathers (Remzi’s dad, Vedat Arpaci) used to say, “Learn beyond the classroom”.'
        for charToSend in arpaciQuote:
            putChar(charToSend)
        putChar(0)    # the solution!!


      def putChar(character):
        global count, putIndex, bufLock, charRemoved, charAdded
        bufLock.acquire()
        while count >= bufsize:
            charRemoved.wait()
        count += 1
```

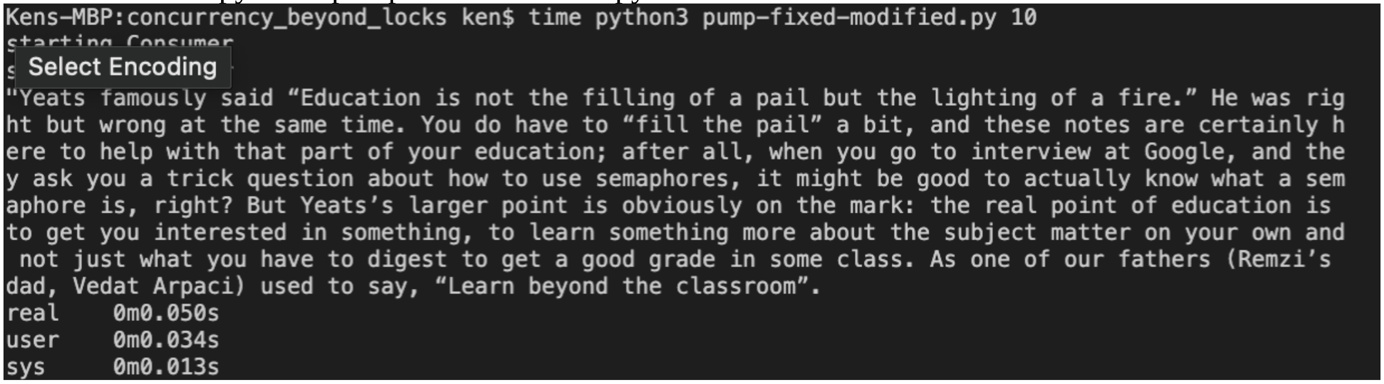


```
    cbuffer[putIndex] = character
    putIndex += 1
    if putIndex == bufsize:
        putIndex = 0
    charAdded.notify()
    bufLock.release()

def pumpConsumer():
    print('starting Consumer')
    while (1):
        for i in range(100):
            c = getChar()
            if c == 0:
                return
            print(c,end="")
        print("")  # newline

def getChar():
    global count, getIndex, bufLock, charRemoved, charAdded
    bufLock.acquire()
    while (count == 0):
        charAdded.wait()
    count -= 1
    c = cbuffer[getIndex]
    getIndex += 1
    if (getIndex == bufsize):
        getIndex = 0
    charRemoved.notify()
    bufLock.release()
    return c

### main ###
if len(sys.argv) != 2:
    print(len(sys.argv))
    print("usage: pump bufferSize")
    exit(1)

bufsize = int(sys.argv[1])
cbuffer = ['x'] * bufsize    # circular buffer; x means uninitialized
count = putIndex = getIndex = 0
bufLock = threading.Lock()
charAdded = Condition(bufLock)
charRemoved = Condition(bufLock)

consumer = threading.Thread(target=pumpConsumer)
consumer.start()

producer = threading.Thread(target=pumpProducer)
```

```
        producer.start()

        producer.join()

```

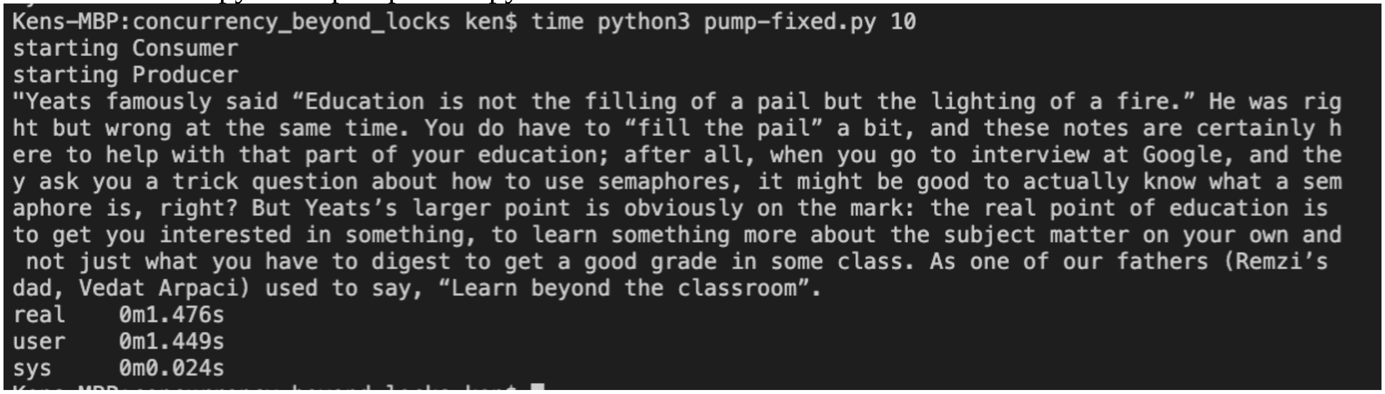
# Output:

Command: time python3 pump-fixed-modified.py 10



```
Kens-MBP:concurrency_beyond_locks ken$ time python3 pump-fixed-modified.py 10
starting Consumer
s
"Yeats famously said “Education is not the filling of a pail but the lighting of a fire.” He was rig
ht but wrong at the same time. You do have to “fill the pail” a bit, and these notes are certainly h
ere to help with that part of your education; after all, when you go to interview at Google, and the
y ask you a trick question about how to use semaphores, it might be good to actually know what a sem
aphore is, right? But Yeats’s larger point is obviously on the mark: the real point of education is
to get you interested in something, to learn something more about the subject matter on your own and
 not just what you have to digest to get a good grade in some class. As one of our fathers (Remzi’s
dad, Vedat Arpaci) used to say, “Learn beyond the classroom”.
real    0m0.050s
user    0m0.034s
sys     0m0.013s
```

Command: time python3 pump-fixed.py 10



```
Kens-MBP:concurrency_beyond_locks ken$ time python3 pump-fixed.py 10
starting Consumer
starting Producer
"Yeats famously said “Education is not the filling of a pail but the lighting of a fire.” He was rig
ht but wrong at the same time. You do have to “fill the pail” a bit, and these notes are certainly h
ere to help with that part of your education; after all, when you go to interview at Google, and the
y ask you a trick question about how to use semaphores, it might be good to actually know what a sem
aphore is, right? But Yeats’s larger point is obviously on the mark: the real point of education is
to get you interested in something, to learn something more about the subject matter on your own and
 not just what you have to digest to get a good grade in some class. As one of our fathers (Remzi’s
dad, Vedat Arpaci) used to say, “Learn beyond the classroom”.
real    0m1.476s
user    0m1.449s
sys     0m0.024s
Kens-MBP:concurrency_beyond_locks ken$
```